Amendments to the Claims:

This listing of claims will replace all prior versions, and listings of claims in the application:

Listing of Claims:

1. (Currently amended) A software program for providing instructions to one or more processors to execute processes on an embedded computing device configured for establishing a network connection with at least one other computing device, comprising:

(a) an operating system layer;

(b) an application framework; [[and]]

(c) a programming environment including a contention locking scheme for setting light object locks, which are handled in user space, and heavy object locks, which are handled at the system level, [[and]] wherein the contention locking scheme is configured to set a light object lock on an initially unlocked object when a first thread attempts to lock the object, and to maintain a light lock on the object when a nested intra-thread lock is attempted by the first thread[[.]];

(d) a stack-based local lock slot structure for addressing stack variables to identify threads; and

(e) a first stack corresponding to a data area of the first thread and a second stack corresponding to a data area of the second thread, the first stack and the second stack being separated by at least a reserved area.

2. (Currently amended) The software program of Claim 1, wherein the contention locking scheme is further configured to set a heavy object lock on the object when [[a]]the second thread attempts to lock the object while the object is lightly locked by the lock attempt by the first thread.

3-4. (Canceled)

5. (Currently amended) The software program of Claim [[4]]2, wherein the contention locking scheme is configured to maintain the light lock when an address difference between a current lock slot of the first thread for the lightly locked object and that of the nested intra-thread locking attempt is determined to be less than the reserved area.

6. (Currently amended) The software program of Claim [[4]]2, wherein the contention locking scheme is configured to set the heavy lock when an address difference between a current lock slot of the first thread for the lightly locked object and that of the locking attempt by the second thread is determined to be greater than the reserved area.

7. (Currently amended) The software program of Claim [[3]]1, wherein the contention locking scheme further includes a lock structure and a lock structure reference in an object header of the object, the lock structure including a lock holder and wait queues.

8. (Canceled)

9. (Currently amended) The software program of Claim [[8]]1, wherein the contention locking scheme is configured to maintain the light lock when an address difference between a current lock slot of the first thread for the lightly locked object and that of the nested intra-thread locking attempt is determined to be less than [[a]]the reserved area ~~which is set at the end of each stack~~.

10. (Currently amended) A software program for providing instructions to one or more processors to execute processes on an embedded computing device configured for establishing a network connection with at least one other computing device, comprising:

(a) an operating system layer;

(b) an application framework; [[and]]

(c) a programming environment; and

(d) ~~including~~ a contention locking scheme for setting light object locks, which are handled in user space, and heavy object locks, which are handled at the system level, ~~and wherein~~ the contention locking scheme is configured to set a light object lock on an initially

unlocked object when a first thread attempts to lock the object, [[and]] to maintain a light lock on the object when a nested intra-thread lock is attempted by the first thread, and to compare an address difference between a current lock slot of the first thread for the lightly locked object and that of the nested intra-thread locking attempt with a size of a reserved area at an end of a stack, the contention locking scheme includes:

(e)~~(d) wherein the contention locking scheme includes~~ a stack-based local lock slot structure for addressing stack variables to identify threads, and

(f)~~(e) wherein the contention locking scheme further includes~~ a lock structure and a lock structure reference in an object header of the object, the lock structure including a lock holder and wait queues.

11. (Currently amended) A method for executing processes with contention-locking which uses light locks, which are handled at the user level, and heavy locks, which are handled at the system level, on an embedded computing device configured for establishing a network connection with at least one other computing device, comprising the steps of:

(a) setting a light object lock, for handling at the user level, on an initially unlocked object when a first thread attempts to lock the object; [[and]]

(b) determining to maintain the light object lock when a nested intra-thread lock is attempted by the first thread[[.]]; and

(c) addressing stack variables to identify threads using a stack-based local lock slot structure, the addressing stack variables to identify threads comprises:

addressing the first thread at a first stack; and

addressing a second thread at a second stack, the second stack being separated from the first stack by at least a reserved area.

12. (Currently amended) The method of Claim 11, further comprising the step of setting a heavy object lock on the object when [[a]]the second thread attempts to lock the object while the object is lightly locked by the lock attempt by the first thread.

13-14. (Canceled)

15. (Currently amended) The method of Claim [[14]]12, wherein the step of determining to maintain the light object lock includes determining that an address difference between a current lock slot of the first thread for the lightly locked object and that of the nested intra-thread locking attempt is less than the reserved area.

16. (Currently amended) The method of Claim [[13]]12, further comprising the step of forming a lock structure and a lock structure reference in an object header of the object, the lock structure including a lock holder and wait queues.

17. (Original) The method of Claim 12, wherein the step of determining to set the heavy lock includes determining that an address difference between a current lock slot of the first thread for the lightly locked object and that of the locking attempt by the second thread is greater than the reserved area.

18. (Canceled)

19. (Currently amended) The method of Claim [[18]]11, wherein the step of determining to maintain the light object lock includes determining that an address difference between a current lock slot of the first thread for the lightly locked object and that of the nested intra-thread locking attempt is less than the reserved area.

20. (Currently amended) The method of Claim [[18]]11, further comprising the step of forming a lock structure and a lock structure reference in an object header of the object, the lock structure including a lock holder and wait queues.